**臺北科技大學資訊工程系**

**107學年度實務專題計畫成果報告**

**基於深度殘差學習理論的改良式快速區域卷積神經網絡之夜間市區車種行人辨識系統**

專題編號：107-CSIE-S003

專題計劃參與人員：104360098 梁皓鈞

104590043 吳建毅

指導教授：陳彥霖 教授

執行期間：106年1學期至107年1學期

摘 要

實務專題名稱：基於深度殘差學習理論的改良式快速區域卷積神經網絡之夜間市區車種行人辨識系統

頁數：46頁

校所別：國立臺北科技大學 資訊工程系學士班

執行期間：106年1學期至107年1學期

專題計劃參與人員：梁皓鈞、吳建毅

指導教授：陳彥霖 教授

關鍵詞：載具辨識、深度學習、黑暗環境、卷積神經網絡、殘差結構、照明不足

不良視野環境下的辨識，一直以來都是一個很大的課題，也是視覺辨識、計算機視覺領域研究的重點之一。不良視野環境包括大雨、霧、夜間、照明不足等。以往一些大量依靠人工設計的特徵抽取及分類器演算法例如HOG + SVM等，在照明不足下往往需要針對黑暗或夜間背景進行處理，可能是一些人工設定的標準化、調整對比度等。但這些方法沒辦法有效地處理整個夜間辨識課題。根據一些前人的研究文獻，夜間辨識往往十分依賴車頭及車尾燈，如果該車輛根本沒有開燈的情況下以及環境照明不足下，辨識能力會大幅下降。在硬體設備起飛的時代，使用深度學習相關的進行辨識已經是一個常態。

大部份用於物件偵測的公開數據集，如PASCAL VOC Challenge 等都是在有足夠環境燈光情況下收集的。本研究假設使用上述類型的數據集所訓練出來的模型在夜間環境下或照明嚴重不足的情況下會有不滿意的表現。另一個研究假設是有著Shortcut Linkages 結構的模型，如殘差網絡 (Residual Networks) 可以解決一些嚴重的問題例如資訊損失 (Information Loss)。本研究提出的解決方案包括收集夜間車種行人數據、標記數據及針對夜間及燈光不足所改良的Faster R-CNN模型用於夜間車種行人辨識系統。本系統最完整版本可以在辨識500 × 375大小的圖片時達到15 - 17 FPS的運算速度，並且在本研究的驗證數據集中達到0.8497 mAP的表現，其中包括了一些市區平常夜間環境照明及嚴重缺乏照明的環境，反映出本研究的系統在多種夜間辨識的能力。

ABSTRACT

Title: Nighttime Vehicle Detection System with Improved Faster R-CNN based on Residual Structure

Pages: 46

School: National Taipei University of Technology

Department: Department of Computer Science and Information Engineering

Time: October, 2018

Project Member(s): Ho Kwan Leung, Jian-Yi Wu

Advisor: Prof. Yen-Lin Chen

Keywords: Vehicle Detection, Deep Learning, Dark Condition, Convolutional Neural Networks, Residual Architecture, Insufficient Lighting

Detection and recognition under bad weather condition or insufficient lighting is a crucial task especially in the field of image detection, recognition, and computer vision. Bad condition includes rainstorm, heavy fog, nighttime and situation of insufficient ambient lighting. Historically, feature extraction and classifiers vastly relying on man-made design such as HOG + SVM need to be altered specifically for the dark or nighttime background with normalization and adjustment of contrast. Nevertheless, these methods can’t efficiently manage the entire nighttime detection task. According to the previous researchs, headlights and taillights of vehicles take a very important role in nighttime vehicle detection. The ability of detection and recognition will suffer from a great decrease if the headlights / taillights are switched off or under insufficient ambient light. Currently, the computing power is enhanced due to the development of hardward, performing detection using deep learning technique is a usual approach.

Most of the public datasets for object detection tasks such as PASCAL VOC Challenge were collected under the conditions with sufficient ambient lighting. We presumed that the models trained by those datasets have unsatisfactory performances in places lacking proper illumination. Another assumption made by our research is that a model with shortcut linkages such as Residual Nets should cope with the problem of serious information loss, especially the objects occupying a small number of pixels and the infrequent existence of crucial features. This study presents the solution including the collection and labeling of nighttime data and the specifically altered system with Faster R-CNN. The system has a processing speed of 15 - 17 FPS for an image size of 500 × 375 and it achieves 0.8497 mAP in our validation segment including city views during nighttime and the situations of extremely inadequate lighting, which shows the ability of detection in various nighttime environments.

目 錄

[摘 要 i](#_Toc525648114)

[ABSTRACT ii](#_Toc525648115)

[目 錄 iv](#_Toc525648116)

[表目錄 vi](#_Toc525648117)

[圖目錄 vii](#_Toc525648118)

[第一章 緒論 1](#_Toc525648119)

[1.1 研究背景與動機 1](#_Toc525648120)

[1.2 研究目的與問題 1](#_Toc525648121)

[1.3 目前研究成果 (科技部大專生研究計劃及研討會論文) 2](#_Toc525648122)

[1.4 報告架構 3](#_Toc525648123)

[第二章 文獻回顧 4](#_Toc525648124)

[2.1 由感知器到深度卷積神經網絡的背景知識 4](#_Toc525648125)

[2.1.1 感知器 (Perceptron) 4](#_Toc525648126)

[2.1.2 神經網絡 (Neural Networks; NN) 5](#_Toc525648127)

[2.1.3 激勵函數 (Activation Function) 6](#_Toc525648128)

[2.1.4 損失函數及代價函數 (Loss Function and Cost Function) 6](#_Toc525648129)

[2.1.5 反向傳播算法 (Back Propagation; BP) 7](#_Toc525648130)

[2.1.6 深度神經網絡 (Deep Neural Networks; DNN) 8](#_Toc525648131)

[2.1.7 卷積神經網絡 (Convolutional Neural Networks; CNN) 8](#_Toc525648132)

[2.1.8 參數隨機初始化 10](#_Toc525648133)

[2.1.9 梯度下降法 (Gradient Descent) 及其變種 11](#_Toc525648134)

[2.1.10 VGG16 12](#_Toc525648135)

[2.1.11 Network in Network 與 Bottleneck 13](#_Toc525648136)

[2.1.12 Batch Normalization (Batch Norm; BN) 14](#_Toc525648137)

[2.1.13 深度殘差網絡 (Deep Residual Networks; ResNet) 15](#_Toc525648138)

[2.2 物件偵測方法 18](#_Toc525648139)

[2.2.1 R-CNN 19](#_Toc525648140)

[2.2.2 Fast R-CNN 20](#_Toc525648141)

[2.2.3 Faster R-CNN 20](#_Toc525648142)

[第三章 研究方法 22](#_Toc525648143)

[3.1 系統架構 22](#_Toc525648144)

[3.2 訓練集資料蒐集與標註 23](#_Toc525648145)

[3.3 訓練集資料增強(Data Augmentation) 24](#_Toc525648146)

[3.4 應用於嵌入式平台之輕量化卷積神經網路 25](#_Toc525648147)

[第四章 實驗結果與分析 31](#_Toc525648148)

[4.1 實驗環境 31](#_Toc525648149)

[4.2 實驗結果 33](#_Toc525648150)

[4.3 文獻比較 37](#_Toc525648151)

[4.4 實驗總結 39](#_Toc525648152)

[第五章 結論與未來工作 41](#_Toc525648153)

[5.1 結論 41](#_Toc525648154)

[5.2 未來工作 41](#_Toc525648155)

[參考文獻 42](#_Toc525648156)

表目錄

圖目錄

[圖1-1 車程行人辨識系統的用途 Tesla Autopilot 自駕車展示 (左)，CoThinking 於印度統計車流量 (中)，Street surveillance 用於道路管制 (右) 1](#_Toc525648157)

[圖1-2 目前已收集的夜間車種行人影像其中兩張 2](#_Toc525648158)

[圖2-1 感知器的組成 4](#_Toc525648159)

[圖2-2 神經網絡的組成 5](#_Toc525648160)

[圖2-3 常用激勵函數 6](#_Toc525648161)

[圖2-4 Convolution的運算 9](#_Toc525648162)

[圖2-5 Max Pooling的運算 9](#_Toc525648163)

[圖2-6 卷積神經網絡示意圖 10](#_Toc525648164)

[圖2-7 Adam 演算法 [14] 11](#_Toc525648165)

[圖2-8 Adam [14] 以 MATLAB 程式編寫 12](#_Toc525648166)

[圖2-9 VGG16的架構 [6] 13](#_Toc525648167)

[圖2-10 Network in Network形式 [15] 14](#_Toc525648168)

[圖2-11 Batch Normalization 的計算 [16] 14](#_Toc525648169)

[圖2-12 Degradation Problem [7] 15](#_Toc525648170)

[圖2-13 Residual building block [7] 16](#_Toc525648171)

[圖2-14 ResNet34 [7] 17](#_Toc525648172)

[圖2-15 詳細ResNet設計 [7] 18](#_Toc525648173)

[圖2-16 ResNet 效果 [7] 18](#_Toc525648174)

[圖2-17 R-CNN [8] 運算過程 19](#_Toc525648175)

[圖2-18 Fast R-CNN [9] 運算過程 20](#_Toc525648176)

[圖2-19 Faster R-CNN [3] 運算過程 21](#_Toc525648177)

[圖2-20 Anchor Boxes 進行Sliding Window [3] 21](#_Toc525648178)

[圖2-21 RoI Pooling [3] 運算過程 21](#_Toc525648179)

1. 第一章 緒論
   1. 研究背景與動機

車種行人辨識系統可用之處十分多。例如圖1-1顯示道路管制、自動駕駛、智慧城市發展、公營停車場停車規劃、採集數據等，是一個發展智慧城市必要的一個城市科學。

|  |
| --- |
| use_of_detection |
| 圖1-1 車程行人辨識系統的用途 Tesla Autopilot 自駕車展示 (左)，CoThinking 於印度統計車流量 (中)，Street surveillance 用於道路管制 (右) |

根據 IBM 提出的智慧城市概觀中發展的多個不同領域中，要提升成為國際智慧城市，本計劃的夜間車種行人辨識系統可以在警政安保、政府與管理機構、建築、交通、環境領域中作出重大貢獻。但目前基於現役的車種行人辨識系統很多都是使用了淺層學習如支持向量機 (SVM) 及非一體成形的演算法，在以往數據稀少及硬體計算能力不足時可以發揮良好的效果。

近年深度學習領域經常大放異彩，大量與影像辨識、物件偵測相關的演算法被提出，這或許是我們城市一個進行全面升級至國際智慧城市的大好時機，研究更加準確更加有效率的物件偵測用於城市管理上有助建立一個更加人性化的城市。針對可能用於夜間車種行人辨識系統，主要原因來自大部份數據都是以有日照或照明作為背景，即大部份數據都是於日間進行收集，因此訓練出的模型都是針對日間的車種行人作出辨識，對於夜間的支援卻十分少，而且夜間需求模型解析能力更高。為了解決相關問題，本專題會同時收集夜間車種行人數據作為訓練模型所用。

* 1. 研究目的與問題

大部份的網上公開數據都是日間或有照明的物體比較多，物件偵測、影像辨識大型數據庫如ImageNet [1]、PASCAL VOC Challenge [2]等都是日間或有照明的物體為主，針對夜間的資料數據十分少，而在夜間的車種行人方面的公開可用數據更加是幾乎沒有。因此本專題研究在作出研究的第一步便是收集夜間市區的車種行人數據。

|  |
| --- |
| sample_two_nighttime_images |
| 圖1-2 目前已收集的夜間車種行人影像其中兩張 |

圖1-2展示的本專題研究收集夜間數據後對數據進行標籤。如之前提及，現時公開的物件偵測及影像辨識數據庫都是於日間或有照明的數據。用這些數據所訓練出來的模型無法良好地提取夜間影像的特徵、線條等用以辨識車種及行人。本專題研究透過使用Faster R-CNN [3]、ILSVRC 2014 [4] 的數據進行預訓練 (Pretrain) 及 PASCAL VOC 2007 [5] 的數據進行訓練後，於夜間影像上有辨識跳動、不穩定的情況。這些很有可能原因來自小訓練及訓練的數據都是日間影像數據，導致其在夜間表現不良。其次原因來自特徵提取解析能力較弱，目前模型多使用 VGG16 [6] 架構模型，是一個在2014年被提出的深度卷積神經網絡模型，但目前已經有其他模型架構與理論被提出且未曾被應用在車種行人辨識上，因此本專題研究採用較新較強特徵提取能力架構進行調整。

夜間市區車種行人辨識系統必要是實時運作的，因此其運作效率一定要高，某種程度上是需要在精準度 (Precision) 及速度上作出取捨 (Trade-off)。但現時硬體愈來愈強的情況下，有足夠的計算能力可以支持做更大更精準的模型。

* 1. 目前研究成果 (科技部大專生研究計劃及研討會論文)

本專題目前研究成果取得科技部大專學生研究計劃，編號為MOST-106-2628-E-027-001-MY3，亦於2018年8月19日在中華民國影像處理與圖形辨別學會 (Chinese Image Processing and Pattern Recognition Society; IPPR) 的第31屆電腦視覺、圖學暨影像處理研討會 (Conference on Computer Vision, Graphics and Image Processing; CVGIP) 進行口頭發表。

* 1. 報告架構

本報告架構分為五個章節，第一章介紹本專題之研究動機、目的與問題；第二章介紹相關背景知識，包含感知器、神經網絡、CNN卷積神經網路模型、Network in Network (NiN)、殘差網絡 (ResNet)、VGG16架構及物件偵測的演化及演算法；第三章介紹本專題研究的研究方法、流程架構、數據標籤及辨識系統的設計與設定；第四章列出本專題的實驗結果與數據，並與其他的CNN模型比較；第五章為本專題之結論與未來工作。

1. 第二章 文獻回顧

本專題採用Faster R-CNN [3] 作為市區夜間車種行人辨識系統的物件偵測演算法。其中，在Faster R-CNN [3] 的特徵提取(Feature Extraction)部份，本專題研究使用VGG16 [6] 及 ResNet101 [7] 作為特徵提取的利害，找尋對於本專題最優良的架構。物件偵測的演算法有很多種基於不同哲學不同根據的發展，其中以region proposal 方式進行偵測的演算法最具代表性的是R-CNN [8]、Fast R-CNN [9] 及Faster R-CNN [3]。本章節將會由感知器 (Perceptron) 開始介紹整個發展深度學習發展概況到物件偵測演算法的演進。

* 1. 由感知器到深度卷積神經網絡的背景知識
     1. 感知器 (Perceptron)

感知器 (Perceptron) 是最早的人工神經元，其結構十分簡單，主要由輸入、權重 (weights) 偏差 (bias) 及激勵函數 (activation function) 組成，如圖2-1。

|  |
| --- |
| **perceptron** |
| 圖2-1 感知器的組成 |

我們可以把感知器視為把每一個資料的特徵乘上一個屬於他自己的權重，把這些都加起來，再加上亦可以視為的偏差值 (bias)，然後送到一個函數，函數的輸出便是感知器的輸出。最原始的激勵函數是Step Function，即

|  |  |
| --- | --- |
|  | (2-1) |

而整個感知器的公式可以視為

|  |  |
| --- | --- |
|  | (2-2) |

可是在這樣的感知器被發明後，科學家不知道該怎樣應用他，也不知道該怎樣訓練他。主要原因是感知器可以模擬成電子電路中的NAND Gate，而NAND Gate則可以模擬出所有的Boolean Logic，這代表了感知器也只不過是其中一種NAND Gate類的操作而已，因此人工神經元的研究進入了寒冬期。

* + 1. 神經網絡 (Neural Networks; NN)

神經網絡 (Neural Networks) 可以視為是由很多個感知器所組成。神經網絡定義上其實只要輸入有輸出，通常是一層或以上 (即只有輸出層，不包括輸入層)，每一層可能有數個神經元，就已經算是神經網絡。以圖2-2為例，這邊有1層輸入層 (3個神經輸入)、1層隱藏層 (5個神經元) 及1層輸出層 (2個神經元)。這邊每一個神經元的操作都像之前所提到的感知器操作一樣。所有權重或偏差一開始都需要隨機初始化。

|  |
| --- |
| neuralnetworks |
| 圖2-2 神經網絡的組成 |

如圖2-2，假設輸入為一筆資料，記成 ，輸入層到隱藏層的權重及偏差為 及 ，隱藏層到輸出層的權重及偏差為 及 ，輸入層到隱藏層的激勵函數與隱藏層到輸出層的激勵函數分別為 與 。如果以向量型式寫法，整個運算的公式可以視為

|  |  |
| --- | --- |
|  | (2-3) |

* + 1. 激勵函數 (Activation Function)

激勵函數不一定只能用先前提到原始的Step Function，也有其他不同類型的激勵函數，現時激勵函數通常是可微分的，原因在於訓練神經網絡需要用到梯度下降法或其變種，其中如果微分等於0則沒法進行訓練，因此Step Function 是沒辦法用在現代的神經網絡中。其他比較常用的激勵函數如圖2-3所顯示。

|  |
| --- |
| 1_ZafDv3VUm60Eh10OeJu1vw |
| 圖2-3 常用激勵函數 |

整體來說現代的激勵函數主要要求是可微分以及連續，且通常以0為準作出變化。不同的激勵函數微分的後數值的大小也會影響到神經網絡訓練的收歛速度及會否發生梯度爆炸或梯度消失。除了上面所提到的，通常現代的神經網絡如果最後是要作出分類，是會使用Softmax 作激勵函數，Softmax實際上是Sigmoid 的變種，其微分與Sigmoid 一樣，在這邊就不一一詳細列出。

* + 1. 損失函數及代價函數 (Loss Function and Cost Function)

損失函數與代價函數的中文名稱常常不固定，因此之後直接稱Loss Function 及 Cost Function 比較準確。Loss Function 通常是指對於描述一個輸出的表現的一個公式。而Cost Function 則是指對於整個神經網絡包括其正則化 (Regularization) 等其他用來表達或者控制神經網絡表現的整個函數，也就是說基本上Loss Function 是被包含在Cost Function 之中。

最常見的Loss Function 是Mean Square Error (MSE) 及 Cross Entropy。一般而言，如果是在做Regression 問題，會用MSE，在分類問題方面則會使用Cross Entropy比較多。MSE與Cross Entropy計算方式分別如下：

|  |  |
| --- | --- |
|  | (2-4) |

|  |  |
| --- | --- |
|  | (2-5) |

在公式(2-4)及(2-5)中，是指一個神經元的輸出，則為Ground-truth Label。這兩者都是可微分而且在目標範圍內都是連續的。假如同一時間訓練的資料不只是一筆，那麼只要把算出來的進行加總再除於該次訓練數據量即可得到該次訓練的平均Loss，也就是該神經網絡的Cost。假如神經網絡的輸出是多個輸出，例如圖2-2有兩個神經元輸出，那麼該神經網絡的Cost 就是把每一個神經元輸出所計算的Loss加總起來再除於加總的數量。

* + 1. 反向傳播算法 (Back Propagation; BP)

要更新一個神經元的權重 (weights) 及偏差 (bias)，可以透過Gradient Methods，把整個問題視為一個最佳化 (Optimization) 問題，使用的方式通常都是梯度下降法 (Gradient Descent) ，如公式(2-6)，或其變種。公式(2-6)中是指Cost Function。主要透過微分的方式求出優化的方向及下降大小。而在神經網絡中，有很多層的神經元，可以透過由Hinton提出的反向傳播算法 [10]，找出更前層的權重所對應的微分，從而用在梯度下降法中進行權重更新。實際上反向傳播算法 [10] 的用法是使用了微積分中的Chain Rule 方式來處理公式(2-6)中的。公式(2-6)是以向量方式表達，則為learning rate，用來控制每次訓練學習的步長。

|  |  |
| --- | --- |
|  | (2-6) |

假如往前推的層數比較多，梯度項透過Chain Rule 也會愈來愈多項相乘，如公式(2-7)，為激勵函數前的數值。如果每一項都比1小，那麼很多項相乘後更新的幅度就會非常小。因此選擇好的激勵函數有助訓練速度。

|  |  |
| --- | --- |
|  | (2-7) |

舉一個實際例子，以圖2-2為例，假設我們稱是指第層的權重且是由第層的第個神經元連接到第層的第個神經元的權重，是指在第層的第個神經元未經激勵函數的數值，是指在第層的第個神經元經過激勵函數後的數值，是指最後一層第個神經元經過激勵函數後的輸出。如果輸入層定義成第0層且整個神經網絡全部的激勵函數都是Sigmoid Function及Loss Function 是使用MSE，使用傳統梯度下降法的更新公式則如公式(2-8)。

|  |  |
| --- | --- |
|  | (2-8) |

* + 1. 深度神經網絡 (Deep Neural Networks; DNN)

基本上深度神經網絡是指層數比較多或者有多重處理的神經網絡。比較傳統的就是傳統神經網絡且有很多層隱藏層。傳統的神經網絡的層被稱為全連接層 (Fully Connected Layer)，意思是每一個神經元都跟下一層全部神經元有權重上的連接，是一個密度十分高的神經網絡。

通常極限會在兩三層隱藏層就已經沒辦法再有什麼大提升了，主要原因是來自Universal Approximation Theorem中所提及，理論上神經網絡只需要一層夠大的隱藏層就已經足夠逼近任何一個Function。因此更多層只是為了不要有Overfitting 的問題以及抽象化。比較現代的深度神經網絡一般都是指深度卷積神經網絡或其他深度神經網絡的架構及變種。

* + 1. 卷積神經網絡 (Convolutional Neural Networks; CNN)

卷積神經網絡是在20世紀60年代時，有科學家在研究貓腦皮層中發現貓在面對不同條紋形狀時神經會有所反應，因此這部份可以應用在計算機視覺中。卷積的概念實際上也常用在一些訊號處理，不過在卷積神經網絡中的卷積後的結果的意義跟訊號處理有一點點不一樣，在訊號處理中，卷積可能是用在映射某一段訊號到另一個訊號，也可以作出放大或濾波的功能。而在計算機視覺中的卷積，很多時候是用來找尋訊號 (或圖片) 中哪一部份的資訊跟卷積核 (Kernel; Filter) 是接近的。如果卷積出來在該部份的數值比較大，就代表該部份的資訊訊號跟卷積核是相近的，也就是我們就會知道在該部份有這樣的訊號特徵。卷積神經的成名主要是在十分有名的數字辨識 [11] 及用在大型圖片辨識比賽中的深度卷積神經網絡 [12] 中大放異彩。

以圖2-4為例，我們可以看到卷積出來的Feature Map 中，由下向上數第2列，由左向右數第2欄的數值4，是整個Feature Map 中數值最大的，回到Input Image 部份，我們可以看到對應的位置的像素數值是跟Filter的數值一模一樣，因此這樣的卷積可以讓我們在圖像中找到一些跟Filter相似的特徵作模型的訓練。該部份的計算為。

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\convolution.png |
| 圖2-4 Convolution的運算 |

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\max pooling.png |
| 圖2-5 Max Pooling的運算 |

池化 (Pooling) 的作用是主要是透過某一些表達可以減少整個數據的大小。比較常用的是最大池化 (Max Pooling) 及平均池化 (Average Pooling)。以最大池化為例，是為了挑出重要資訊的同時減少數據大小，在圖2-5中可以看到，在Feature Map 中以每大小的視窗滑動，找出視窗中最大的數值並且保留，這樣做就可以在減少數據大小我同時保留重要資訊。而平均池化其實就是把視窗中的所有數值取平均。

在卷積層及池化層中，都會有Strides 跟 Padding的概念。Strides 是指窗口滑動的步幅，如果Strides 是2，則代表窗口每次滑動兩個像素。而Padding 則是為了解決窗口滑動沒辦法有所有部份都在圖像的邊際及角落進行操作，因此會在整個圖片外圍補上一些0，讓圖片變大後再進行卷積或池化，這樣在經過卷積或池化後，Feature Map 的大小可以不縮小，且可以對圖像的邊際像素進行全面的操作。

通常我們常說的是 “Same Padding”或 “Valid Padding”，前者是指Padding 的大小剛好令Feature Map 的大小與輸入圖片的大小相同，後者則是指平常的操作。一般而言，假設輸入圖片的大小為，Filter 大小為，Strides 為，Feature Map 的大小可以以公式(2-9)作計算。

|  |  |
| --- | --- |
|  | (2-9) |

傳統的卷積神經網絡通常是經由數個卷積層及池化層後，最後把輸出展平再接上一兩層全連接層再輸出。常見的卷積神經網絡表達圖如圖2-6。

|  |
| --- |
|  |
| 圖2-6 卷積神經網絡示意圖 |

* + 1. 參數隨機初始化

如章節2.1.2中提及，神經網絡的所有參數都是需要隨機初始化，如果全部一樣會出現Symmetric Break 的問題，也就是對稱問題。隨機初始化最傳統的就是使用Uniform 或 Normal (Gaussian) 分佈作初始化。隨機初始化實際上對神經網絡是十分重要的，初始化的形態會嚴重影響到神經網絡在收歛時是收歛到多深的局部最低點 (Local Minima)。因此在這方面也有不少相應的研究，目前對於ReLU的激勵函數有著俗稱Xavier Initialization [13] 的隨機初始化方法，經過研究可以加速使用ReLU的神經網絡收歛，更加可以比較容易收歛到比較深的局部最低點。Xavier Initialization [13] 的計算如公式(2-10)，可以看到他的基本思想是保持輸入和輸出的方差 (Variance) 一致。除此之外，還有其他基於這種初始化方式的變種，但都離不開這種初始化的基本思想。

|  |  |
| --- | --- |
|  | (2-10) |

* + 1. 梯度下降法 (Gradient Descent) 及其變種

神經網絡經由把整個問題視為一個函數最佳化進行訓練，是目前神經網絡主流的訓練方式。如章節2.1.5中提及，大體離不開梯度下降法這種Gradient Search 的方式或其變種。但傳統的梯度下降法存在不少可以改良的地方，如公式(2-6)所顯示，可以看到整個梯度都由同一個常數進行縮放，控制步幅大小。因此有不少研究如何讓整個計算在不同的方向可以有不同的步幅控制，同時也以物理的角度思考如何讓整個優化過程有像動量 (Momentum) 般的方式，藉此來突破一些局部最低點。

有不少相應的改良都被提出，目前最知名最常用的Optimizer為 Adam [14]，其演算法如圖2-7。以比較直觀的方式理解整個Adam的算法，以MATLAB方式寫出如圖2-8，此程式來源為本人其他專案項目的更新權重部份。

|  |
| --- |
|  |
| 圖2-7 Adam 演算法 [14] |

|  |
| --- |
|  |
| 圖2-8 Adam [14] 以 MATLAB 程式編寫 |

* + 1. VGG16

VGG系列的卷積神經網絡架構是由University of Oxford 的 Visual Geometry Group 所開發的。VGG16 [6] 是一個16層卷積神經網絡，成名於 2014年 ILSVRC 比賽中取得亞軍。該神經網絡告訴了我們大小的filter 有足夠能力解析特徵，也告訴了我們一個規規矩矩的架構的確可以有不錯的辨識能力。VGG16 [6] 的架構如圖2-9所展示。

|  |
| --- |
|  |
| 圖2-9 VGG16的架構 [6] |

基本上可以看到VGG16 [6] 所有的filter的大小都是，而filter的數量是以2倍作增加，池化全部都是Max Pooling，是一個中規中矩、有系統的架構。這樣的架構在當時算是一個十分深的卷積神經網絡，也表現十分優秀，辨識率是當時比賽中十分耀眼的，這證明了其卷積部份的設定有著良好的特徵解析能力。

* + 1. Network in Network 與 Bottleneck

Network in Network (NiN) 如同其名字所指，網絡中的網絡。他的大概意思是在卷積層跟卷積層之間，可以接上一些小網絡，從而達到增加非線性特性效果，如圖2-10所展示。主要實現是以 Convolution的方式，也就是說以一個大小為 的Filter進行卷積。這樣聽起來好像很奇怪，因為卷積的目的是為了一個範圍中提取特徵，但根本沒有範圍可言。實際上卷積意義十分重大，其主要運用後的意義為

1. 增加非線性特性
2. 透過減少Channel 量達到某種程度的Dimensionality Reduction，反之亦然
3. 前一層的輸出如果是，使用卷積作用與全連接層是一樣的
4. 以Bottleneck形式出現，可以降維減少運算量後再升維回復資訊
5. 融合多張Feature Map的資訊

|  |
| --- |
|  |
| 圖2-10 Network in Network形式 [15] |

這樣的設計愈來愈多在近年的深度卷積神經網絡架構中出現，特別是在本專題研究中所使用到的 ResNet101 [7] 中大量出現，目的為降維減少運算量後再升維回復資訊。

* + 1. Batch Normalization (Batch Norm; BN)

Batch Normalization (BN) [16] 的計算如圖2-11。BN層是被用在神經層中已作線性組合後，激勵函數前。基本上可以視為一種縮放分佈的手法，主要思想為Normalize 該層的線性組合後得到一個平均為0，方差為1的分佈。這樣的做法是屬於論文中的Internal Covariate Shift [16]，可以藉此一定程度上解決當使用以0為主要變化的激勵函數所產生的梯度消失的問題。

|  |
| --- |
|  |
| 圖2-11 Batch Normalization 的計算 [16] |

而圖2-11中的及作用是提及神經網絡有能力可以回復到沒有BN操作的情況。這樣的好處是讓神經網絡自己決定在該層的BN [16] 操作是否需要。因此透過及可以一定程度上把整個分佈進行調整到對神經網絡是好的表現。因此及不是計算出來的，而是一個可以透過反向傳播法 [10] 進行訓練的兩個參數。這兩個參數在多次訓練疊代下，可以泛化整層分佈的變化，因此不是直接計算的。

值得一提的是，BN [16] 只是一種做法，實際上用在哪一個維度是有不同意義的。在全連接層中，通常是作用在數據量的維度，而在卷積層中，通常是作用在Channel 的維度。

* + 1. 深度殘差網絡 (Deep Residual Networks; ResNet)

神經網絡愈深，就會面臨愈多問題，包括因為反向傳播法 [10] 中會有大量的微分的相乘，容易導致梯度消失或爆炸，也有可能會有Overfitting 的問題。其中一個問題叫Degradation [7]，中文直譯叫優化退化，由Kaiming He 在深度殘差網絡 [7] 論文中提出。這個問題既不是Overfitting所致，也未必是單純是梯度消失所致。詳細論文中沒有提及，但只是有說明有相關現象，實際上這個現象有可能是由於太多卷積令某些微小特徵出現資訊損失。該現象如圖2-12所展示。

|  |
| --- |
|  |
| 圖2-12 Degradation Problem [7] |

Kaiming He 提出的解決方式叫Residual的架構 [7]，可以有效在大幅添加層數情況下不受Degradation 的問題所影響。一個最小的Residual單元如圖2-13所展示。基本上可以理解成有某一層的輸出再去下一層時分開成兩條路，一條是照原本做法，進行線性組合後經激勵函數後輸出，接下來再進行線性組合且暫時未經激勵函數。另一條路則是直接與第一條路的結尾相加，然後再一起經過激勵函數。以數學方式理解就是，更準確的說明應該是。

|  |
| --- |
|  |
| 圖2-13 Residual building block [7] |

先從這個最單純的單一building block 進行理解，用回先前所使用的符號，假如神經網絡中間的部份，，是無用的，神經網絡大可以在、、或訓練成全0的矩陣，也就是說完全完全無作用的一層半。這樣的話在相加的部份就會變成，而預設的激勵函數為ReLU，亦即是，因此再經由ReLU後仍然是。簡單理解可以認為是讓神經網絡決定因應不同的特徵自行決定哪些需要更多的卷積層進行處理，哪些已經足夠了。

標準的ResNet34 [7]，也就是34層版本的殘差網絡如圖2-14所顯示。基本上在中間大部份地方都有加入殘差的架構。圖2-15顯示Kaiming He 所提出的所有ResNet不同層數的版本的設計。基本上可以看到由一開始的一個卷積及 Max Pooling作神經網絡一開頭的處理，接著是由4大殘差區塊作運算，接下來再到Global Average Pooling後全連接層進行分類輸出。Global Average Pooling 實際上就是全圖的平均池化，其窗口大小就是與Feature Map 大小一樣，因此池化後的Feature Map 的大小便是。值得注意的是，整個神經網絡池化的部份只有一開始的 Max Pooling及最後的Global Average Pooling，在神經網絡中間完全沒有使用任何池化的操作，而是在圖2-15中的卷積區塊conv3\_x、conv4\_x及conv5\_x的一開始使用Strides = 2 的卷積方式令Feature Map 大小減少一半。

根據該論文所提及，ResNet18及ResNet34不需要使用到Bottleneck架構。而其他ResNet50、ResNet101及ResNet152都有使用到，主要作用就如章節2.1.11所提到的功用。而有關在進行殘差後前後的Feature Map 大小不一樣，相加的方式有三種，分別是補zero-padding、線性映射用於增加大小及對所有shortcuts進行線性映射。作者提出他們認為線性映射用於增加大小的方式是最好的，也是目前標準使用ResNet的方式。殘差的效果如圖2-16所顯示，以ResNet的方式增加層數不會有Degradation Problem，而且辨識能力有更明顯的提升。有關使用Batch Normalization [16] 的部份，根據論文所指，ResNet中每一層都有使用，而使用方式如章節2.1.12所提及。

|  |
| --- |
|  |
| 圖2-14 ResNet34 [7] |

|  |
| --- |
|  |
| 圖2-15 詳細ResNet設計 [7] |

|  |
| --- |
|  |
| 圖2-16 ResNet 效果 [7] |

* 1. 物件偵測方法

傳統的物件偵測方式大多數都依賴多重人工設計的運算步驟。例如在特徵設定及提取方面會使用到Haar、HOG等方式。之後再使用Sliding Window 的方式在特徵圖上滑動，把每個顯示在Sliding Window 上的矩陣給傳統的分類器例如SVM等進行分類。這種傳統的方法一來沒有特定提取潛在物件的策略、時間複雜度高。另一方面是人工特徵的設計未必是最好的。隨著深度學習的出現，這些方式慢慢被深度學習的方式取代，現時最常見的方式有兩種，分別為

1. Region Proposal 類的物件偵測演算法
2. 基於Regression 方式的物件偵測演算法

本專題研究使用的是Region Proposal 類方式，因此以下文獻回顧以相關演算法為主。

* + 1. R-CNN

R-CNN [8] 是由Ross Girshick 於2014年在其論文中提出的Region Proposal 方式的演算法，屬於使用深度學習作物件偵測的開山之作。R-CNN [8] 的計算過程基本上可以分為4大步驟，如圖2-17所展示。分別是

1. 一張圖經Selective Search 生成1K到2K個候選區域
2. 對於每個候選區域使用卷積神經網絡的卷積部份進行特徵提取
3. 把提取出來的特徵送到每一類的SVM分類器
4. 使用Regressor 修正每個Bounding Box 的位置

|  |
| --- |
| C:\Users\Henry\Desktop\RCNN.JPG |
| 圖2-17 R-CNN [8] 運算過程 |

在第一步驟中，使用Selective Search 方法從一張圖像生成約2K個候選區域。Search Search 實際上是一種Hierarchical Grouping 的方式，使用透過觀察顏色紋理的分割手段把圖像分割成小區域，再透過合併可能性最高的兩個區域，重複合併直到整張圖片合併完成。曾經存在過的區域就是所謂的候選區域。

在第二步驟中，會把從第一步驟中所選出的候選區域剪出，並且進行一定程度的縮放，根據論文會縮放到大小的圖片。這部份實際上如果本身物件的形態與縮放後的形態不太一樣，會嚴重影響辨識能力。因此不一定是進行縮放，也有可能是以Padding 的方式進行補足。再把處理後的候選區域送到卷積神經網絡中的卷積部份進行特徵抽取，成為多張Feature Maps。

第三步驟，會把Feature Maps 送到針對每一個類別進行決定的SVM進行分類。分類決定後再送到第四步驟。

第四步驟則是使用Regressor 把Bounding Box 的座標進行精修。

要注意的是，R-CNN [8] 不是一個一體成形的演算法，需要分部份進行訓練，因此訓練時間十分久，而且運算也十分慢，不太能達到實時計算的效果。另一個問題是訓練所需要的空間比較大，主要是因為整個演算法需要獨立分開訓練。

* + 1. Fast R-CNN

Fast R-CNN [9] 是由Ross Girshick 提出的針對R-CNN [8] 的改良。大幅提升了物件偵測的速度，大幅減少了訓練時間，運算時間也由原本十分久改良到可以算是實時運算的速度。Fast R-CNN [9] 的運算過程如圖2-18，步驟為

1. 一張圖經由Selective Search 生成候選區域
2. 對於每個候選區域使用卷積神經網絡的卷積部份進行特徵提取，並且在較深的部份的接上RoI Pooling
3. 經由RoI Pooling 進行池化及結合候選區域的座標進行Bounding Box 定位
4. 同時送到Softmax 進行分類及bbox regressor 進行精修框的動作

|  |
| --- |
| C:\Users\Henry\Desktop\fast rcnn.png |
| 圖2-18 Fast R-CNN [9] 運算過程 |

Fast R-CNN [9] 最大貢獻是在把分類的部份改為使用全連接神經網絡經由Softmax 進行分類以及RoI Pooling 的概念。這些概念都會延續在接下來更強物件偵測演算法。有關RoI Pooling 的概念會在下一章節 Faster R-CNN [3] 部份進行說明。

* + 1. Faster R-CNN

Faster R-CNN [3] 是一個真正把 R-CNN系列的演算法變成一體成形 end to end 的演算法，所有運算可以完全在GPU上完成，大幅提升了速度。過程中也使用了不少比較新的想法及結構，包括Convolutional Sliding Window、Region Proposal Network (RPN)、Anchor Box機制。整個運算流程如圖2-19，分別為

1. 圖片直接經由卷積神經網絡的卷與部份進行特徵抽取
2. 經由RPN進行候選物件提取
3. 結合先前的Feature Map 送到RoI Pooling進行池化
4. 同時送到Softmax 進行分類及bbox regressor 進行精修框的動作

|  |  |
| --- | --- |
|  | https://i2.read01.com/uploads/0EnCXgPdsF.jpg |
| 圖2-19 Faster R-CNN [3] 運算過程 | 圖2-20 Anchor Boxes 進行Sliding Window [3] |

Faster R-CNN 的卷積部份是與RPN共享的，這樣可以大幅減少運算時間，這樣的概念是來自Convolutional Sliding Window。RPN 實際上是一個Sliding Window 的方式進行選取候選區域的，如圖2-20顯示，對圖片進行Sliding Window 後個別送到卷積部份進行特徵抽取其實是與直接用卷積部份對整個圖片進行卷積後在Feature Map 上進行Sliding Window 是一樣的，但後者明顯運算時間減少許多。而Sliding Window 的窗口是使用預定的數個Anchor Boxes 在Feature Map 上滑動。

為了解決Proposal 大小不同的問題，RoI Pooling 由Fast R-CNN [9] 就已經被提出，也沿用到Faster R-CNN [3] 。為了解決經過Pooling 後出現的失真問題，RoI Pooling 是基於Feature Map 出來的Proposal 的大小來決定Pooling 的大小，這樣就可以不受到Proposal 大小不同而影響，運算過程如圖2-21顯示。

|  |
| --- |
| C:\Users\Henry\Desktop\資策會深度學習\roi pooling.png |
| 圖2-21 RoI Pooling [3] 運算過程 |

1. 第三章 研究方法
   1. 系統架構

本論文之系統架構圖如圖3-1，分為兩個部分，分別為訓練階段與偵測階段，在訓練階段，首先準備好標註完成的7個類別樣本，分別是Person、Car、Motorbike、Truck、Bus、Bike、Van，並將建立好的卷積神經網路進行訓練，訓練完成後並將其學習的所有權重存檔，在偵測階段時，讀入影像後，對整張影像進行縮放為網路模型可接受的輸入大小，再將已訓練好的網路模型讀入並輸入影像進行物件的偵測與分類，判斷影像中是否存在事先定義好的7個物件。

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\system flowchart.png |
| 圖3-1、系統架構圖 |

* 1. 訓練集資料蒐集與標註

為了提升數據收集的效率，本論文實驗透過實車上路、行走時或行駛汽機車時拍攝，並標記出目標物座標與類別如圖3-2，輸出xml之標記檔案如圖3-3。在數據的收集過程中，需要決定收集到的數據是好或壞。若拍攝影像中之目標物過小或模糊，則視為不好的數據，如圖3-4；反之，若為清晰影像，且目標物件有一定大小以上，則為可利用之數據，如圖3-5。目標物件在影像中的些微傾斜，並不影響其數據品質，清晰與否為較重要之指標。

|  |  |
| --- | --- |
| 圖3-2、物件標記示意圖 | |
|  | |
| 圖3-3、輸出之xml檔案格式 | |
|  |  |
| 圖3-4、品質不好的數據 | 圖3-5、品質較好的數據 |

將標記檔案直接可視化於訓練集影像如圖3-6。按照這7個類別進行標記，並可以對物件之標記框進行修改及編輯。

|  |  |
| --- | --- |
|  |  |
| 圖3-6、標記好之訓練集影像 | |

* 1. 訓練集資料增強(Data Augmentation)

由於夜間資料集與目前公開的資料集比較起來而言是相對稀少的，為了解決資料集稀少的問題，會將影像經過旋轉、調整大小、比例尺寸，或者改變亮度色溫、翻轉等影像處理，一張影像經過處理後，我們人眼仍能辨識出來是相同的影像，但是對機器來說是完全不同的新影像，因此藉由資料增強將既有的影像資料予以修改變形，創造出更多不同的影像來讓機器學習，以彌補資料量不足的問題。本論文實驗使用的資料增強方法主要有下列幾點影像處理方法。

1. 旋轉(Rotation)：隨機旋轉影像。
2. 翻轉(Flip)：沿著水平方向翻轉或者垂直方向翻轉影像。
3. 縮放(Zoom)：按照等比例放大或者縮小影像。
4. 平移(Shift)：採用隨機的方式指定平移範圍和平移步長，沿水平或垂直方向進行平移，改變影像內容的位置。
5. 尺度變換(Scale)：對影像按照指定的尺度進行放大或縮小。
6. 對比度變換(Contrast)：在圖像的HSV顏色空間，改變飽和度(S)和亮度(V)分量，維持色調H不變，對每個像素的S和V分量進行指數運算(指數範圍在0.25到4之間)，以增加光照亮度變化。
7. 雜訊擾動(Noise)：對影像加入椒鹽雜訊和高斯雜訊。
   1. 應用於嵌入式平台之輕量化卷積神經網路

為了減少傳統卷積運算本身計算量龐大之問題，主要使用SqueezeNet卷積神經網路中之Fire module，建立輕量化卷積神經網路，具體實現細節將在以下條列式詳細列出

1. 在Fire module中，Expand層採用了混合卷積核和，其strides均為1，對於卷積核，其輸出Feature map與原始一樣大小，但由於要和得到的Feature map做concat，所以卷積進行了padding=”same”的操作
2. 採用了Max pooing層，即Pool size大小為，strides = 2
3. 訓練採用Adam加快模型的收斂速度與優化，初始學習速率為0.001
4. Fire module中所有卷積核使用Xavier Initialization[14]做權重的隨機生成，激活函數採用ReLU再做輸出，可以看到卷積核是有能力提取線條，而且在經過多次訓練後，會慢慢找到抓取特徵最好的卷積核數值，如圖3-7，(b)(c)(d)之Feature map分別為由式(3-1)、式(3-2)、式(3-3)之卷積核卷積之結果。
5. 由於本論文實驗之卷積神經網路使用了較多層的Fire module與較多的Filter數量如表3-1，因此在模型的最後使用額外深度可分離卷積層(Depthwise Separable Convolution)來提取額外的候選框，以取代文獻[9] SSD中連接VGG16之後的額外卷積層。

|  |  |  |
| --- | --- | --- |
|  | | |
| (a)輸入之影像 | | |
|  |  |  |
| (b)輸出之Feature map 1 | (c)輸出之Feature map 2 | (d)輸出之Feature map 3 |
| 圖3-7、使用Xavier Initialization初始化卷積核進行學習 | | |

|  |  |
| --- | --- |
|  | (3-1) |

|  |  |
| --- | --- |
|  | (3-2) |

|  |  |
| --- | --- |
|  | (3-3) |

表3-1、Fire Module之卷積網路模型

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Stride | Filter Shapes | Output Size |
| Conv1 | 2 |  |  |
| Pool1 | 2 |  |  |
| Fire1 | 1 | ；E1；E3 |  |
| Concate1 | | | |
| Fire2 | 1 | ；E1；E3 |  |
| Concate2 | | | |
| Fire3 | 1 | ；E1；E3 |  |
| Concate3 | | | |
| Pool2 | 2 |  |  |
| Fire4 | 1 | ；； |  |
| Concate4 | | | |
| Fire5 | 1 | ；； |  |
| Concate5 | | | |
| Fire6 | 1 | ；； |  |
| Concate6 | | | |
| Pool3 | 2 |  |  |
| Fire7 | 1 | ；； |  |
| Concate7 | | | |
| Fire8 | 1 | ；； |  |
| Concate8 | | | |
| Fire9 | 1 | ；； |  |
| Concate9 | | | |
| Fire10 | 1 | ；； |  |
| Concate10 | | | |
| Pool4 | 2 |  |  |
| Fire11 | 1 | ；； |  |
| Concate11 | | | |
| Pool5 | 2 |  |  |
| Concate12 | | | |
| Fire12 | 1 | ；； |  |
| Concate12 | | | |
| SeparableConv1 | 2 |  |  |
| SeparableConv2 | 1 |  |  |
| SeparableConv3 | 1 |  |  |
| SeparableConv4 | 1 |  |  |

Depthwise Separable Convolution為一種對卷積神經網路的新卷積運算結構，以降低 卷積神經網路所產生的運算成本。該方式主要將原先的卷積運算方式拆成兩個部分，分別為深度卷積(Depthwise Convolutions) 與逐點卷積(Pointwise Convolutions)，在不影響輸出結構的情況下減少運算量。

Depthwise Convolutions 層則是先對輸入層的每個channel 建立一個  的 Filter如圖3-8，之後對該 channel 所有的資料進行卷積運算。假如輸入層有 M 個 channel，則會建立 M 個 Filter， 分別對資料進行運算得出與輸出層相同寬高的中間層如圖3-9。

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\depthwise conv filter.png |
| 圖3-8、Depthwise Convolutions Filter |

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\Depthwise conv.png |
| 圖3-9、Depthwise Convolutions運算 |

在每個channel進行過 Depthwise Convolutions 的步驟後，下一步是對每個點的所有channel進行 Pointwise Convolutions。先對每個channel 建立一個大小為  的 Filter後 (M 為輸入層的 channel 數)，如圖3-10，將輸入層的所有點進行卷積運算。假如輸出層有 N 個 channel，則會建立 N 個 的 Filter，之後將每個 Filter 對輸入層進行運算後可得到大小為 的輸出層( 為輸出層的長寬)。該結果與傳統 卷積層輸出結構是相同的，如圖3-11。

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\pointwise conv filter.png |
| 圖3-10、Pointwise Convolutional Filter |

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\pointwise conv.png |
| 圖3-11、Pointwise Convolutions 運算 |

由前面的討論可知 Depthwise Convolutions 部分的計算量為式(3-4)

|  |  |
| --- | --- |
|  | (3-4) |

而 Pointwise Convolutions 則為式(3-5)

|  |  |
| --- | --- |
|  | (3-5) |

因此 Depthwise Separable Convolution 的總計算量可由 Depthwise 與 Pointwise 兩部份相加得到式(3-6)

|  |  |
| --- | --- |
|  | (3-6) |

與傳統卷積的計算量進行比較，可得到下列式(3-7)之結果

|  |  |
| --- | --- |
|  | (3-7) |

由上述可知道原先卷積中的 Filter Size 與輸出的channel越大，使用Depthwise Separable Convolution 方式所減少的計算量會更多。以一個大小 3×3 的 Filter來說計算量可減少到原先的1/9 倍，對效能有非常大的幫助。

在預測與分類之卷積層中，仿照SSD的方式，在主要的卷積網路中的後面幾層卷積層與額外增加的深度可分離卷積層皆使用得卷積核產生一系列的物件框Prior box，來做物件的位置預估與物件類別的分類，用於位置預估與物件類別的分類之卷積層如圖3-12。

|  |
| --- |
| C:\Users\Henry\Desktop\碩士論文\論文圖片\fire ssd.png |
| 圖3-12，用於預測與分類之Feature map |

在訓練方式中，網路模型以Adam為優化方法，初始的learning rate 為0.001，訓練過程當loss無明顯降低時，會將learning rate降低為原來的0.2倍，此外，在訓練過程中若超過10個epoch的loss無降低，則會判定模型loss已收斂而提早結束訓練，圖3-13為經過217個epoch，每個epoch為1000 step之模型loss曲線圖。

|  |
| --- |
|  |
| 圖3-13、網路模型訓練之收斂曲線 |

1. 第四章 實驗結果與分析
   1. 實驗環境

本論文實驗所提出的系統主要實作的程式語言為Python3，其中於Python3第三方套件中會使用到TensorFlow[15]、Keras[16]作為深度學習框架、Tensorboard、NumPy、BeautifulSoup作為數據處理及可視化工具，主要是使用到Keras作為前端API，而以Tensorflow作為後端之Framework。版本方面近期TensorFlow 有改版更新，為了系統可執行性及一致性，我們採用TensorFlow 1.6版本作為基準，而Keras則為2.1.6。

在訓練模型方面，運用NVIDIA GeForce GTX1080Ti 顯示卡[17]用作訓練模型使用。深度學習的研究需要有想法後實作，再透過實驗驗證。之所以會使用這麼大量的運算硬體資源是因為做影像大量深度卷積，需要有足夠的運算能力支持。

因此本論文實驗分別於個人電腦與NVIDIA Jetson TX2嵌入式平台[18]進行實驗，個人電腦設備如表4-1：

表4-1、PC實驗平台之規格

|  |  |  |
| --- | --- | --- |
| 類型 | 項目 | 規格說明 |
| 硬體環境 | CPU | Intel® Core™ i7-7700K CPU @ 4.20GHz × 8 |
| GPU | GeForce® GTX 1080 Ti |
| RAM | DDR4 24GB |
| 軟體環境 | 作業系統 | Ubuntu 16.04.4 LTS (Xenial Xerus) 64bit |
| 程式開發工具 | Visual Studio Code |
| 程式語言 | Python3.5 |
| 深度學習框架 | TensorFlow 1.6  Keras 2.1.6 |
| 相依函式庫 | OpenCV 3.4.1[19]  CUDA 9.0[20]  cuDNN 7.0[21] |

NVIDIA Jetson TX2為人工智慧運算專用且的嵌入式開發平台，如圖4-1。適合於需要在低耗能環境中使用高運算效能的應用程式，具備完整的 NVIDIA Jetpack SDK開發套件，可用於深度學習的程式庫、電腦視覺、GPU 運算等多媒體處理，其規格與本論文實驗環境如表4-2

在環境設置上我們由於是使用TensorFlow 1.6 作為主要開發版本，相關的圖形運算環境及套件如CUDA、cuDNN等版本都必需要處理好。CUDA我們使用9.0而cuDNN則使用7.0的版本。TensorFlow 方面由於使用NVIDIA Jetson TX2，在CUDA運算上需要使用到顯示卡加速，因此本論文實驗是安裝TensorFlow GPU，但為了最佳化配合讓效能可以發揮到最好，需要從TensorFlow原始碼進行完全重新編譯再進行安裝，有了TensorFlow作為底層Framework後，Keras只需要從最方便的Python3-pip 套件進行安裝即可

由於NVIDIA Jetson TX2 是屬於嵌入式裝置而不是一個訓練深度學習用的裝置，因此我們在訓練的過程不會使用到NVIDIA Jetson TX2，而是在每一次訓練完成後，除了在一般電腦上即使測試成果及問題外，也會每一次都導入到TX2進行測試。

|  |
| --- |
| C:\Users\Henry\Desktop\P_20180707_145713.jpg |
| 圖4-1、NVIDIA Jetson TX2 |

表4-2、NVIDIA Jetson TX2實驗平台之規格

|  |  |  |
| --- | --- | --- |
| 類型 | 項目 | 規格說明 |
| 硬體環境 | CPU | HMP Dual Denver 2/2 MB L2 +  Quad ARM® A57/2 MB L2 |
| GPU | NVIDIA Pascal™, 256 CUDA cores |
| RAM | DDR4 8 GB 128 bit |
| 軟體環境 | 作業系統 | Ubuntu 16.04.4 LTS (Xenial Xerus) 64bit |
| 程式語言 | Python3.5 |
| 深度學習框架 | TensorFlow 1.6  Keras 2.1.6 |
| 相依函式庫 | OpenCV 3.4.1  CUDA 9.0  cuDNN 7.0 |

* 1. 實驗結果

本節將展示本論文實驗的結果與數據。系統評估之方法如表表4-3、表4-4，定義事件類型、偵測率、漏報率、誤報率、正確率，來評估本論文系統方法

表4-3、事件定義

|  |  |
| --- | --- |
| 事件分類 | 事件說明 |
| 物件正確數(E1) | 定義之事件(標的物)存在，且系統偵測到之事件(標的物) |
| 漏抓數(E2) | 定義之事件(標的物)存在，且系統未偵測到之事件(標的物) |
| 誤抓數(E3) | 定義之事件(標的物)不存在，且系統偵測到之事件(標的物) |

表4-4、數據分析定義

|  |  |
| --- | --- |
| 結果 | 計算方式 |
| 偵測率 | E1 / (E1 + E2) |
| 漏報率 | E2 / (E1 + E2) |
| 誤報率 | E3 / (E1 + E3) |
| 正確率 | E1 / (E1 + E2 + E3) |

實驗用的行車紀錄器為GoPro Hero6 Black，在一般市區速限為時速40至50公里，為了避免拍攝的影像因移動造成的模糊，將影像錄製設定為1080p、120 FPS，而在實際偵測階段會轉換為30 FPS之輸入影像串流，圖4-2到圖4-5是為實驗用的夜間行車紀錄影片之場景截圖，包含車前、車左側、車右側之視角。

|  |  |
| --- | --- |
|  |  |
| 圖4-2、車前場景-1截圖 | 圖4-3、車前場景-2截圖 |
|  |  |
| 圖4-4、車左側場景-1截圖 | 圖4-5、車右側場景-1截圖 |

圖4-6到圖4-17是為實驗用的夜間行車紀錄影片之執行成果圖，表4-5為本實驗於個人電腦上的執行結果數據統計

表4-5、夜間場景偵測數據統計(個人電腦實驗結果)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 影片  名稱 | 物件  總數 | 物件  正確數 | 漏抓數 | 誤抓數 | 偵測率 | 正確率 | 平均單張影像耗費時間 |
| 車前場景-1 | 20249 | 20060 | 189 | 7 | 99.0% | 99.0% | 0.012 s |
| 車前場景-2 | 7477 | 7328 | 149 | 4 | 98.0% | 98.0% | 0.0098 s |
| 車左側場景-1 | 4465 | 4151 | 314 | 6 | 93.0% | 92.8% | 0.0088 s |
| 車右側場景-1 | 4341 | 4170 | 171 | 5 | 96.0% | 96.0% | 0.0092 s |

|  |  |
| --- | --- |
|  |  |
| 圖4-6、車前場景-1偵測結果(一) | 圖4-7、車前場景-1偵測結果(二) |
|  |  |
| 圖4-8、車前場景-2偵測結果(一) | 圖4-9、車前場景-2偵測結果(二) |
|  |  |
| 圖4-10、車左側場景-1偵測結果(一) | 圖4-11、車左側場景-1偵測結果(二) |
|  |  |
| 圖4-12、車左側場景-1偵測結果(三) | 圖4-13、車左側場景-1偵測結果(四) |
|  |  |
| 圖4-14、車右側場景-1偵測結果(一) | 圖4-15、車右側場景-1偵測結果(二) |
|  |  |
| 圖4-16、車右側場景-1偵測結果(三) | 圖4-17、車右側場景-偵測結果(四) |

圖4-18到圖4-25是為實驗用的夜間行車紀錄影片之執行成果圖，表4-5為本實驗於NVIDIA Jetson TX2嵌入式平台上的執行結果數據統計

表4-6、夜間場景偵測數據統計(NVIDIA Jetson TX2實驗結果)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 影片  名稱 | 物件  總數 | 物件  正確數 | 漏抓數 | 誤抓數 | 偵測率 | 正確率 | 平均單張影像耗費時間 |
| 車前場景1 | 20249 | 19905 | 344 | 7 | 98.3% | 98.3% | 0.16 s |
| 車前場景2 | 7477 | 7292 | 185 | 4 | 97.5% | 97.5% | 0.10 s |
| 車左側場景1 | 4465 | 4151 | 314 | 6 | 93.0% | 92.8% | 0.10 s |
| 車右側場景1 | 4341 | 4170 | 171 | 5 | 96.0% | 96.0% | 0.10 s |

|  |  |
| --- | --- |
|  |  |
| 圖4-18、車前場景-1偵測結果(一) | 圖4-19、車前場景-1偵測結果(二) |
|  |  |
| 圖4-20、車前場景-2偵測結果(一) | 圖4-21、車前場景-2偵測結果(二) |
|  |  |
| 圖4-22、車左側場景-1偵測結果(一) | 圖4-23、車左側場景-1偵測結果(二) |
|  |  |
| 圖4-24、車右側場景-1偵測結果(一) | 圖4-25、車右側場景-1偵測結果(二) |

* 1. 文獻比較

在文獻[9]的網路模型訓練上，使用與本論文實驗相同的訓練集與測試集，影像大小為，5191張訓練影像中包含了7種物件，且訓練至模型收斂為止，測試時也依照本論文實驗之評估之方法，統計模型之表現，表4-7為文獻[9]於個人電腦上的執行結果數據。表4-9為文獻[9]與本論文實驗方法之比較結果，可以看出本方法可達到與文獻[9]相同之準確率，並且有9成以上之準確率，而平均單張影像耗費時間可從0.023 s提升至0.009 s。

此外也將本論文實驗之模型與文獻[7]進行比較，由於其使用的是RPN來提取Proposal，所以準確度會比較高，而速度上卻明顯降低許多，表4-8為文獻[7]於個人電腦上的執行結果數據，表4-10為文獻[7]與本論文實驗方法之比較結果，平均單張影像耗費時間可從0.067 s提升至0.009 s。

表4-7、文獻[9]夜間場景偵測數據統計(個人電腦實驗結果)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 影片  名稱 | 物件  總數 | 物件  正確數 | 漏抓數 | 誤抓數 | 偵測率 | 正確率 | 平均單張影像耗費時間 |
| 車前場景1 | 20249 | 20060 | 189 | 7 | 99.0% | 99.0% | 0.025 s |
| 車前場景2 | 7477 | 7328 | 149 | 4 | 98.0% | 98.0% | 0.023 s |
| 車左側場景1 | 4465 | 4161 | 304 | 6 | 93.2% | 93.0% | 0.023 s |
| 車右側場景1 | 4341 | 4179 | 162 | 5 | 96.3% | 96.2% | 0.023 s |

表4-8、文獻[7]夜間場景偵測數據統計(個人電腦實驗結果)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 影片  名稱 | 物件  總數 | 物件  正確數 | 漏抓數 | 誤抓數 | 偵測率 | 正確率 | 平均單張影像耗費時間 |
| 車前場景1 | 20249 | 20127 | 122 | 10 | 99.4% | 99.3% | 0.066 s |
| 車前場景2 | 7477 | 7357 | 120 | 8 | 98.4% | 98.3% | 0.066 s |
| 車左側場景1 | 4465 | 4174 | 291 | 6 | 93.5% | 93.4% | 0.067 s |
| 車右側場景1 | 4341 | 4197 | 144 | 6 | 96.7% | 96.5% | 0.068 s |

表4-9、文獻[9]網路模型準確率與速度比較表(個人電腦實驗結果)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 文獻[9] | | 本論文方法 | |
| 影片  名稱 | 正確率 | 平均單張影像耗費時間 | 正確率 | 平均單張影像耗費時間 |
| 車前場景1 | 99.0% | 0.025 s | 99.0% | 0.012 s |
| 車前場景2 | 98.0% | 0.023 s | 98.0% | 0.0098 s |
| 車左側場景1 | 93.0% | 0.023 s | 92.8% | 0.0088 s |
| 車右側場景1 | 96.2% | 0.023 s | 96.0% | 0.0092 s |

表4-10、文獻[7]網路模型準確率與速度比較表(個人電腦實驗結果)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 文獻[7] | | 本論文方法 | |
| 影片  名稱 | 正確率 | 平均單張影像耗費時間 | 正確率 | 平均單張影像耗費時間 |
| 車前場景1 | 99.3% | 0.066 s | 99.0% | 0.012 s |
| 車前場景2 | 98.3% | 0.066 s | 98.0% | 0.0098 s |
| 車左側場景1 | 93.4% | 0.067 s | 92.8% | 0.0088 s |
| 車右側場景1 | 96.5% | 0.067 s | 96.0% | 0.0092 s |

* 1. 實驗總結

從實驗數據可知，要達到網路模型的輕量化，進行卷積的時候需要將通道的卷積與空間的卷積進行分離，才可降低運算量，由於文獻[9]的主網路架構為VGG16，訓練完成的網路模型將近300MB，而本論文實驗訓練出來之網路模型只有78.6MB，不僅可達到與文獻[9]幾乎相同的準確率，在個人電腦的偵測速度上也從原來的40 FPS左右提升至100 FPS；此外文獻[9]由於網路模型過大並無法在NVIDIA Jetson TX2上運行，而本論文實驗的模型可以相同的準確率且10 FPS左右的速度實現在NVIDIA Jetson TX2上。

而文獻[7]的主網路架構也為VGG16，但訓練完成的網路模型達1.1GB，網路模型過大也無法在NVIDIA Jetson TX2上運行；而本論文實驗訓練出來之網路模型與其比較準確率，下降的範圍落在1%，在個人電腦的偵測速度上則從原來的15 FPS左右提升至100 FPS。

1. 第五章 結論與未來工作
   1. 結論

本論文實驗以夜間車載或街道車種行人辨識為研究範疇，提出之輕量化網路模型，結合影像串流並可以在NVIDIA Jetson TX2嵌入式開發板上運行，準確度可達到9成以上，偵測速度在個人電腦平台上平均偵測一張影像的時間約為0.009 s，並且在NVIDIA Jetson TX2平均偵測一張一張影像的時間約為0.10s，奠定將來在移植至其他平台，如車載平台、穿戴式平台等提供一個穩固的基礎，使得在未來應用上的範圍更加廣泛。

* 1. 未來工作

本系統在未來發展上可分為短期與長期兩個部分：

(1) 短期改進部分：

改善對小物件的偵測能力，目前的偵測演算法，為了加快速度，會將輸入影像以固定大小進行偵測，不會進行縮放，但在實際行車影像的應用中，行人或車輛可能會從進距離逐漸遠離，物件會從大慢慢變小，所以進行縮放或是融合每一層地Feature map特徵可能可以改善偵測小物件的效能。

(2) 長期改善部分：

目前用來預測物件位置的卷積層都是固定取某幾層的卷積層卷積結果進行物件Bounding box的提取，未來可以朝輸出之Feature map的數值來得知物件在那些卷積層的卷積結果特徵會消失，並以此機制來推測需要取那些卷積層來進行Bounding box的提取，避免提取之物件出現無效特徵的問題。

加入測距模組，將與危險物件的距離顯示使車主知道，且根據物件距離、人的反應時間、時速與完整煞車時間等等因素進行統整，評估偵測之有效距離，減少意外的發生。

1. 參考文獻
2. 衛生福利部. *焦點新聞 - 103年國人死因統計結果*. Available: <http://www.mohw.gov.tw/news/531349778>.
3. B. Frohlich, M. Enzweiler, and U. Franke, "Will this Car Change the Lane?-Turn Signal Recognition in the Frequency Domain," IEEE Intelligent Vehicles Symposium Proceedings, 2014.
4. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.
5. J. R. R. Uijlings, K. E. A. v. d. Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for objectrecognition,” International Journal of Computer Vision, vol. 104, no. 2, 2013.
6. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, IEEE Conference on Computer Vision and Pattern Recognition, 2014.
7. R. Girshick, “Fast R-CNN,” Proceedings of International Conference on Computer Vision, 2015.
8. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN : Towards real-time object detection with region proposal networks,” Proceedings of Advances in Neural Information Processing Systems, 2015.
9. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016.
10. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD : Single shot multibox detector”, Proceedings of European Conference on Computer Vision, 2016.
11. F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, “SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and < 0.5MB Model Size”, arXiv 16.,2016
12. François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions”,arXiv16.,2016
13. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”,arXiv17.,2017
14. Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, “ ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices”,arXiv17.,2017
15. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Proceedings of Machine Learning Research, vol. 9, pp. 249-256, 2010.
16. Tensorflow, https://www.tensorflow.org/versions/r1.6/
17. Keras, https://keras.io/
18. NVIDIA Geforce gtx 1080ti, http://www.geforce.com.tw/graphics-cards/geforce/pascal/tw/gtx-1080-ti
19. NVIDIA Jetson TX2, https://www.nvidia.com/zh-tw/autonomous-machines/embedded-systems-dev-kits-modules/
20. OpenCV, https://opencv.org/opencv-3-4-1.html
21. NVIDIA CUDA, https://developer.nvidia.com/cuda-toolkit
22. NVIDIA cuDNN, https://developer.nvidia.com/cudnn